

# Efficiently Learning a Detection Cascade with Sparse Eigenvectors

Chunhua Shen, Sakrapee Paisitkriangkrai, and Jian Zhang, *Senior Member, IEEE*

**Abstract**—Real-time object detection has many applications in video surveillance, teleconference and multimedia retrieval *etc.* Since Viola and Jones [1] proposed the first real-time AdaBoost based face detection system, much effort has been spent on improving the boosting method. In this work, we first show that feature selection methods other than boosting can also be used for training an efficient object detector. In particular, we introduce Greedy Sparse Linear Discriminant Analysis (GSLDA) [2] for its conceptual simplicity and computational efficiency; and slightly better detection performance is achieved compared with [1]. Moreover, we propose a new technique, termed Boosted Greedy Sparse Linear Discriminant Analysis (BGS LDA), to efficiently train a detection cascade. BGS LDA exploits the sample re-weighting property of boosting and the class-separability criterion of GSLDA. Experiments in the domain of highly skewed data distributions, *e.g.*, face detection, demonstrates that classifiers trained with the proposed BGS LDA outperforms AdaBoost and its variants. This finding provides a significant opportunity to argue that AdaBoost and similar approaches are not the only methods that can achieve high classification results for high dimensional data in object detection.

**Index Terms**—Object detection, AdaBoost, asymmetry, greedy sparse linear discriminant analysis, feature selection, cascade classifier.

## I. INTRODUCTION

**R**EAL-TIME object detection such as face detection has numerous computer vision applications, *e.g.*, intelligent video surveillance, vision based teleconference systems and content based image retrieval. Various detectors have been proposed in the literature [1], [3], [4]. Object detection is challenging due to the variations of the visual appearances, poses and illumination conditions. Furthermore, object detection is a *highly-imbalanced* classification task. A typical natural image contains many more negative background patterns than object patterns. The number of background patterns can be 100,000 times larger than the number of object patterns. That means, if one wants to achieve a high detection rate, together with a low false detection rate, one needs a specific classifier. The cascade classifier takes this imbalanced distribution into consideration [5]. Because of the huge success of Viola and Jones' real-time AdaBoost based face detector [1], a lot of incremental

work has been proposed. Most of them have focused on improving the underlying boosting method or accelerating the training process. For example, AsymBoost was introduced in [5] to alleviate the limitation of AdaBoost in the context of highly skewed example distribution. Li *et al.* [6] proposed FloatBoost for a better detection accuracy by introducing a backward feature elimination step into the AdaBoost training procedure. Wu *et al.* [7] used forward feature selection for fast training by ignoring the re-weighting scheme in AdaBoost. Another technique based on the statistics of the weighted input data was used in [8] for even faster training. KLBoost was proposed in [9] to train a strong classifier. The weak classifiers of KLBoost are based on histogram divergence of linear features. Therefore in the detection phase, it is not as efficient as Haar-like features. Notice that in KLBoost, the classifier design is separated from feature selection. In this work (part of which was published in preliminary form in [10]), we propose an improved learning algorithm for face detection, dubbed Boosted Greedy Sparse Linear Discriminant Analysis (BGS LDA).

Viola and Jones [1] introduced a framework for selecting discriminative features and training classifiers in a cascaded manner as shown in Fig. 1. The cascade framework allows most non-face patches to be rejected quickly before reaching the final node, resulting in fast performance. A test image patch is reported as a face only if it passes tests in all nodes. This way, most non-face patches are rejected by these early nodes. Cascade detectors lead to very fast detection speed and high detection rates. Cascade classifiers have also been used in the context of support vector machines (SVMs) for faster face detection [4]. In [11], soft-cascade is developed to reduce the training and design complexity. The idea was further developed in [12]. We have followed Viola and Jones' original cascade classifiers in this work.

One issue that contributes to the efficacy of the system comes from the use of AdaBoost algorithm for training cascade nodes. AdaBoost is a forward stage-wise additive modeling with the weighted exponential loss function. The algorithm combines an ensemble of weak classifiers to produce a final strong classifier with high classification accuracy. AdaBoost chooses a small subset of weak classifiers and assign them with proper coefficients. The linear combination of weak classifiers can be interpreted as a decision hyper-plane in the weak classifier space. The proposed BGS LDA differs from the original AdaBoost in the following aspects. Instead of selecting decision stumps with minimal weighted error as in AdaBoost, the proposed algorithm finds a new weak learner that maximizes the class-separability criterion.

NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. X X.

C. Shen is with NICTA, Canberra Research Laboratory, Locked Bag 8001, Canberra, ACT 2601, Australia, and also with the Australian National University, Canberra, ACT 0200, Australia (e-mail: chunhua.shen@nicta.com.au).

S. Paisitkriangkrai and J. Zhang are with NICTA, Neville Roach Laboratory, Kensington, NSW 2052, Australia, and also with the University of New South Wales, Sydney, NSW 2052, Australia (e-mail: {paul.pais, jian.zhang}@nicta.com.au).



Fig. 1. A cascade classifier with multiple nodes. Here a circle represents a node classifier. An input patch is classified as a target only when it passes tests at each node classifier.

As a result, the coefficients of selected weak classifiers are updated repetitively during the learning process according to this criterion.

Our technique differs from [7] in the following aspects. [7] proposed the concept of Linear Asymmetric Classifier (LAC) by addressing the asymmetries and *asymmetric node learning* goal in the cascade framework. Unlike our work where the features are selected based on the Linear Discriminant Analysis (LDA) criterion, [7] selects features using AdaBoost/AsymBoost algorithm. Given the selected features, Wu *et al.* then build an optimal linear classifier for the node learning goal using LAC or LDA. Note that similar techniques have also been applied in neural network. In [13], a nonlinear adaptive feed-forward layered network with linear output units has been introduced. The input data is nonlinearly transformed into a space in which classes can be separated more easily. Since LDA considers the number of training samples of each class, applying LDA at the output of neural network hidden units has been shown to increase the classification accuracy of two-class problem with unequal class membership. As our experiments show, in terms of feature selection, the proposed BGSLDA methods is superior than AdaBoost and AsymBoost for object detection.

The key contributions of this work are as follows.

- We introduce GSLDA as an alternative approach for training face detectors. Similar results are obtained compared with Viola and Jones' approach.
- We propose a new algorithm, BGSLDA, which combines the sample re-weighting schemes typically used in boosting into GSLDA. Experiments show that BGSLDA can achieve better detection performances.
- We show that feature selection and classifier training techniques can have different objective functions (in other words, the two processes can be separated) in the context of training a visual detector. This offers more flexibility and even better performance. Previous boosting based approaches select features and train a classifier simultaneously.
- Our results confirm that it is beneficial to consider the highly skewed data distribution when training a detector. LDA's learning criterion already incorporates this imbalanced data information. Hence it is better than standard AdaBoost's exponential loss for training an object detector.

The remaining parts of the paper are structured as follows. In Section II-A, the GSLDA algorithm is introduced as an alternative learning technique to object detection problems. We then discuss how LDA incorporates imbalanced data

information when training a classifier in Section II-B. Then, in Sections II-C and II-D, the proposed BGSLDA algorithm is described and the training time complexity is discussed. Experimental results are shown in Section III and the paper is concluded in Section IV.

## II. ALGORITHMS

In this section, we present alternative techniques to AdaBoost for object detection. We start with a short explanation of the concept of GSLDA [14]. Next, we show that like AsymBoost [5], LDA is better at handling asymmetric data than AdaBoost. We also propose the new algorithm that makes use of sample re-weighting scheme commonly used in AdaBoost to select a subset of relevant features for training the GSLDA classifier. Finally, we analyze the training time complexity of the proposed method.

### A. Greedy Sparse Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) can be cast as a generalized eigenvalue decomposition. Given a pair of symmetric matrices corresponding to the between-class ( $S_b$ ) and within-class covariance matrices ( $S_w$ ), one maximizes a class-separability criterion defined by the generalized Rayleigh quotient:

$$\max_w \frac{w^T S_b w}{w^T S_w w}. \quad (1)$$

The optimal solution of a generalized Rayleigh quotient is the eigenvector corresponding to the maximal eigenvalue. The sparse version of LDA is to solve (1) with an additional sparsity constraint:

$$\text{Card}(w) = k, \quad (2)$$

where  $\text{Card}(\cdot)$  counts the number of nonzero components, *a.k.a.* the  $\ell_0$  norm.  $k \in \mathbb{Z}^+$  is an integer set by a user. Due to this sparsity constraint, the problem becomes non-convex and NP-hard. In [14], Moghaddam *et al.* presented a technique to compute optimal sparse linear discriminants using branch and bound approach. Nevertheless, finding the exact global optimal solutions for high dimensional data is infeasible. The algorithm was extended in [2], with new sparsity bounds and efficient matrix inverse techniques to speed up the computation time by 1000 $\times$ . The technique works by sequentially adding the new variable which yields the maximum eigenvalue (forward selection) until the maximum number of elements are selected or some predefined condition is met. As shown in [2], for two-class problem, the computation can be made very efficient as the only finite eigenvalue  $\lambda_{\max}(S_b, S_w)$  can be computed in closed-form as  $\mathbf{b}^T S_w^{-1} \mathbf{b}$  with  $S_b = \mathbf{b} \mathbf{b}^T$  because in this case  $S_b$  is a rank-one matrix.  $\mathbf{b}$  is a column vector. Therefore, the computation is mainly determined by the inverse of  $S_w$ . When a greedy approach is adopted to sequentially find the suboptimal  $w$ , a simple rank-one update for computing  $S_w^{-1}$  significantly reduces the computation complexity [2]. We have mainly used forward greedy search in this work. For forward greedy search, if  $l$  is the current subset of  $k$  indices and  $m = l \cup i$  for candidate  $i$  which is not in  $l$ . The new augmented

inverse  $(S_w^m)^{-1}$  can be calculated in a fast way by recycling the last step's result  $(S_w^l)^{-1}$ :

$$(S_w^m)^{-1} = \begin{bmatrix} (S_w^l)^{-1} + a_i \mathbf{u}_i \mathbf{u}_i^\top & -a_i \mathbf{u}_i \\ -a_i \mathbf{u}_i & a_i \end{bmatrix}, \quad (3)$$

where  $\mathbf{u}_i = (S_w^l)^{-1} S_{w,li}$  with  $(li)$  indexing the  $l$  rows and  $i$ -th column of  $S_w$  and  $a_i = 1/(S_{w,ii} - S_{w,li}^\top \mathbf{u}_i)$  [15], [2].

Note that we have experimented with other sparse linear regression and classification algorithms, *e.g.*,  $\ell_1$ -norm linear support vector machines,  $\ell_1$ -norm regularized log-linear models, *etc.* However, the major drawback of these techniques is that they do not have an *explicit* parameter that controls the number of features to be selected. The trade-off parameter (regularization parameter) only controls the degree of sparseness. One has to tune this parameter using cross-validation. Also  $\ell_1$  penalty methods often lead to sub-optimal sparsity [16]. Hence, we have decided to apply GSLDA, which makes use of greedy feature selection and the number of features can be predefined. It would be of interest to compare our method with  $\ell_1$ -norm induced sparse models [17].

The following paragraph explains how we apply GSLDA classifier [2] as an alternative feature selection method to classical Viola and Jones' framework [1].

Due to space limit, we omit the explanation of cascade classifiers. Interested readers should refer to [1], [7] for details. The GSLDA object detector operates as follows. The set of selected features is initialized to an empty set. The first step (lines 4 – 5) is to train weak classifiers, for example, decision stumps on Haar features.<sup>1</sup> For each Haar-like rectangle feature, the threshold that gives the minimal classification error is stored into the lookup table. In order to achieve maximum class separation, the output of each decision stump is examined and the decision stump whose output yields the maximum eigenvalue is sequentially added to the list (line 7, step (1)). The process continues until the predefined condition is met (line 6).

The proposed GSLDA based detection framework is summarized in Algorithm 1.

### B. Linear Discriminant Analysis on Asymmetric Data

In the cascade classifiers, we would prefer to have a classifier that yields high detection rates without introducing many false positives. Binary variables (decision stump outputs) take the Bernoulli distribution and it can be easily shown that the log likelihood ratio is a linear function. In the Bayes sense, linear classifiers are optimum for normal distributions with equal covariance matrices. However, due to its simplicity and robustness, linear classifier has shown to perform well not only for normal distributions with unequal covariance matrices but also non-normal distributions. A linear classifier can be written

<sup>1</sup>We introduce nonlinearity into our system by applying decision stump learning to raw Haar feature values. By nonlinearly transforming the data, the input can now be separated more easily using simple linear classifiers.

Note that any classifiers can be applied here. We also use LDA on covariance features for human detection as described [18]. For the time being, we focus on decision stumps on Haar-like features. We will give details about covariance features later.

**Algorithm 1** The training procedure for building a cascade of GSLDA object detector.

---

**Input:**

- A positive training set and a negative training set;
- A set of Haar-like rectangle features  $h_1, h_2, \dots$ ;
- $D_{\min}$ : minimum acceptable detection rate per cascade level;
- $F_{\max}$ : maximum acceptable false positive rate per cascade level;
- $F_{\text{target}}$ : target overall false positive rate;

```

1 Initialize:  $i = 0$ ;  $D_i = 1$ ;  $F_i = 1$ ;
2 while  $F_{\text{target}} < F_i$  do
3    $i = i + 1$ ;  $f_i = 1$ ;
4   foreach feature do
5     Train a weak classifier (e.g., a decision stump
       parameterized by a threshold  $\theta$ ) with the smallest error
       on the training set;
6   while  $f_i > F_{\max}$  do
7     1. Add the best weak classifier (e.g., decision stump)
       that yields the maximum class separation;
8     2. Lower classifier threshold such that  $D_{\min}$  holds;
9     3. Update  $f_i$  using this classifier threshold;
10   $D_{i+1} = D_i \times D_{\min}$ ;  $F_{i+1} = F_i \times f_i$ ; and remove correctly
      classified negative samples from the training set;
11  if  $F_{\text{target}} < F_i$  then
12    Evaluate the current cascaded classifier on the negative
      images and add misclassified samples into the negative
      training set;

```

**Output:**

- A cascade of classifiers for each cascade level  $i = 1, \dots$ ;
- Final training accuracy:  $F_i$  and  $D_i$ ;

---

as

$$F(\mathbf{x}) = \begin{cases} +1 & \text{if } \sum_{t=1}^n w_t h_t(\mathbf{x}) + \theta \geq 0; \\ -1 & \text{otherwise,} \end{cases} \quad (4)$$

where  $h(\cdot)$  defines a function which returns binary outcome,  $\mathbf{x}$  is the input image features and  $\theta$  is an optimal threshold such that the minimum number of examples are misclassified. In this paper, our linear classifier is the summation of decision stump classifiers. By central limit theorem, the linear classifier is close to normal distribution for large  $n$ .

The asymmetric goal for training cascade classifiers can be written as a trade-off between false acceptance rate  $\varepsilon_1$  and false rejection rate  $\varepsilon_2$  as

$$r = \varepsilon_1 + \mu \varepsilon_2, \quad (5)$$

where  $\mu$  is a trade-off parameter. The objective of LDA is to maximize the projected between-class covariance matrix (distance between the mean of two classes) and minimize the within-class covariance matrix (total covariance matrix). The selected weak classifier is guaranteed to achieve this goal. Having large projected mean difference and small projected class variance indicates that the data can be separated more easily and, hence, the asymmetric goal can also be achieved more easily. On the other hand, AdaBoost minimizes symmetric exponential loss function that does not guarantee high detection rates with little false positives [5]. The selected features are therefore no longer optimal for the task of rejecting negative samples.

Another way to think of this is that AdaBoost sets initial positive and negative sample weights to  $0.5/N_p$  and  $0.5/N_n$  ( $N_p$  and  $N_n$  is the number of positive samples and negative samples). The prior information about the number of samples in each class is then *completely lost* during training. In contrast, LDA takes the number of samples in each class into consideration when solving the optimization problem, *i.e.*, the number of samples is used in calculating the between-class covariance matrix ( $S_B$ ). Hence,  $S_B$  is the weighted difference between class mean and sample mean.

$$S_B = \sum_{c_i} N_{c_i} (\mu_{c_i} - \bar{x})(\mu_{c_i} - \bar{x})^T, \quad (6)$$

where  $\mu_{c_i} = N_{c_i}^{-1} \sum_{j \in c_i} x_j$ ;  $\bar{x} = N^{-1} \sum_j x_j$ ;  $N_{c_i}$  is the number of samples in class  $c_i$  and  $N$  is the total number of samples. This extra information minimizes the effect of imbalanced data set.

In order to demonstrate this, we generate an artificial data set similar to one used in [5]. We learn a classifier consisting of 4 linear classifiers and the results are shown in Fig. 2. From the figure, we see that the first weak classifier (#1) selected by both algorithms are the same since it is the only linear classifier with minimal error. AdaBoost then re-weights the samples and selects the next classifier (#2) which has the smallest weighted error. From the figure, the second weak classifier (#2) introduces more false positives to the final classifier. Since most positive samples are correctly classified, the positive samples' weights are close to zero. AdaBoost selects the next classifier (#3) which classifies all samples as negative. Therefore it is clear that all but the first weak classifier learned by AdaBoost are poor because it tries to balance positive and negative errors. The final combination of these classifiers are not able to produce high detection rates without introducing many false positives. In contrast to AdaBoost, GSLDA selects the second and third weak classifier (#2, #3) based on the maximum class separation criterion. Only the linear classifier whose outputs yields the maximum distance between two classes is selected. As a result, the selected linear classifiers introduce much less false positives (Fig. 2).

In [5], Viola and Jones pointed out the limitation of AdaBoost in the context of highly skewed example distribution and proposed a new variant of AdaBoost called AsymBoost which is experimentally shown to give a significant performance improvement over conventional boosting. In brief, the sample weights were updated before each round of boosting with the extra exponential term which causes the algorithm to gradually pay more attention to positive samples in each round of boosting. Our scheme based on LDA's class-separability can be considered as an alternative classifier to AsymBoost that also takes asymmetry information into consideration.

### C. Boosted Greedy Sparse Linear Discriminant Analysis

Before we introduce the concept of BGSLDA, we present a brief explanation of boosting algorithms. Boosting is one of the most popular learning algorithms. It was originally designed for classification problems. It combines the output of many weak classifiers to produce a single strong learner.

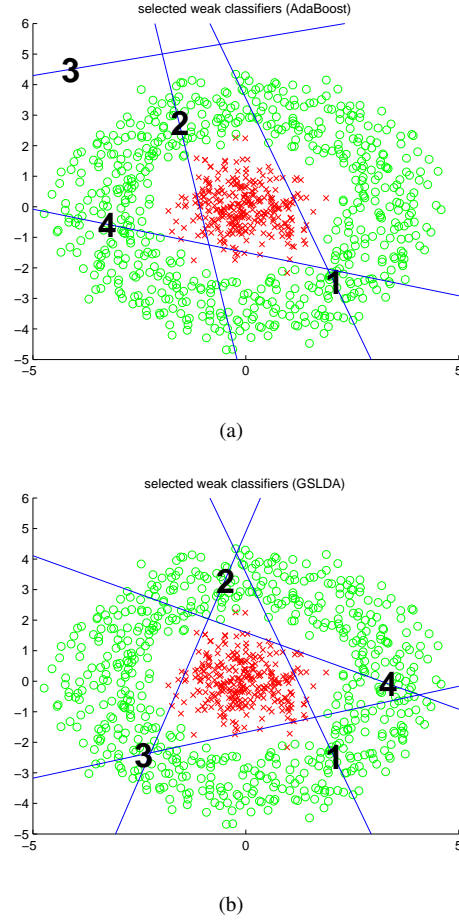


Fig. 2. Two examples on a toy data set: (a) AdaBoost classifier; (b) GSLDA classifier (forward pass).  $\times$ 's and  $\circ$ 's represent positive and negative samples, respectively. Weak classifiers are plotted as lines. The number on the line indicates the order in which weak classifiers are selected. AdaBoost selects weak classifiers for attempting to balance weighted positive and negative error. Notice that AdaBoost's third weak classifier classifies all samples as negative due to the very small positive sample weights. In contrast, GSLDA selects weak classifiers based on the maximum class separation criterion. We see that four weak classifiers of GSLDA model the positives well and most of the negative are rejected.

Weak classifier is defined as a classifier with accuracy on the training set greater than average. There exist many variants of boosting algorithms, *e.g.*, AdaBoost (minimizing the exponential loss), GentleBoost (fitting regression function by weighted least square methods), LogitBoost (minimizing the logistic regression cost function) [19], LPBoost (minimizing the Hinge loss) [20], [21], *etc.* All of them have an identical property of sample re-weighting and weighted majority vote. One of the widely used boosting algorithm is AdaBoost [22]. AdaBoost is a greedy algorithm that constructs an additive combination of weak classifiers such that the exponential loss

$$L(y, F(x)) = \exp(-yF(x))$$

is minimized. Here  $x$  is the labeled training examples and  $y$  is its label;  $F(x)$  is the final decision function which outputs the decided class label. Each training sample receives a weight  $u_i$  that determines its significance for training the next weak classifier. In each boosting iteration, the value of  $\alpha_t$  is

computed and the sample weights are updated according to the exponential rule. AdaBoost then selects a new hypothesis  $h(\cdot)$  that best classifies updated training samples with minimal classification error  $e$ . The final decision rule  $F(\cdot)$  is a linear combination of the selected weak classifiers weighted by their coefficients  $\alpha_t$ . The classifier decision is given by the sign of the linear combination

$$F(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{N_w} \alpha_t h_t(\mathbf{x})\right),$$

where  $\alpha_t$  is a weight coefficient;  $h_t(\cdot)$  is a weak learner and  $N_w$  is the number of weak classifiers. The expression of the above equation is similar to an expression used in dimensionality reduction where  $F(\mathbf{x})$  can be considered as the result of linearly projecting the random vector  $\mathbf{F}_i$  onto a one dimensional space along the direction of  $\alpha$ .

In previous section, we have introduced the concept of GSLDA in the domain of object detection. However, decision stumps used in GSLDA algorithm are learned *only once* to save computation time. In other words, once learned, an optimal threshold, which gives smallest classification error on the training set, remains unchanged during GSLDA training. This speeds up the training process as also shown in forward feature selection of [7]. However, it limits the number of decision stumps available for GSLDA classifier to choose from. As a result, GSLDA algorithm fails to perform at its best. In order to achieve the best performance from the GSLDA classifier, we propose to extend decision stumps used in GSLDA training with sample re-weighting techniques used in boosting methods. In other words, each training sample receives a weight and the new set of decision stumps are trained according to these sample weights. The objective criterion used to select the best decision stump is similar to the one applied in step (1) in Algorithm 1. Note that step (3) in Algorithm 2 is introduced in order to speed up the GSLDA training process. In brief, we remove decision stumps with weighted error larger than  $e_k + \varepsilon$  where  $e_k = \frac{1}{2} - \frac{1}{2}\beta_k$ ,  $\beta_k = \max(\sum_{i=1}^N u_i y_i h_t(x_i))$  and  $N$  is the number of samples,  $y_i$  is the class label of sample  $x_i$ ,  $h_t(x_i)$  is the prediction of the training data  $x_i$  using weak classifier  $h_t$ . The condition used here has connection with the dual constraint of the soft margin LPBoost [20]. The dual objective of LPBoost minimizes  $\beta$  subject to the constraints

$$\sum_{i=1}^N u_i y_i h_t(x_i) \leq \beta, \forall t,$$

and

$$\sum_{i=1}^N u_i = 1, 0 \leq u_i \leq \text{const}, \forall i.$$

As a result, the sample weights  $u_i$  is the most pessimistic one. We choose decision stumps with weighted error smaller than  $e_k + \varepsilon$ . These decision stumps are the ones that perform best under the most pessimistic condition.

Given the set of decision stumps, GSLDA selects the stump that results in maximum class separation (step (4)). The sample weights can be updated using different boosting algorithm (step (5)). In our experiments, we use AdaBoost [1] re-weighting scheme (BGS LDA - scheme 1).

$$D_i^{(t+1)} = \frac{u_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z^{(t+1)}}, \quad (7)$$

**Algorithm 2** The training algorithm for building a cascade of BGS LDA object detector.

---

```

1 while  $F_{\text{target}} < F_i$  do
2    $i = i + 1$ ;
3    $f_i = 1$ ;
4   while  $f_i > F_{\text{max}}$  do
5     1. Normalize sample weights  $\mathbf{u}$ ;
6     2. Train weak classifiers  $h(\cdot)$  (e.g., decision stumps by
7       finding an optimal threshold  $\theta$ ) using the training set
8       and sample weights;
9     3. Remove those weak classifiers with weighted error
10      larger than  $e_k + \varepsilon$  (section II-C);
11     4. Add the weak classifier whose output yields the
12      maximum class separation;
13     5. Update sample weights  $\mathbf{u}$  in the AdaBoost manner
14      (Eq. (7)) or AsymBoost manner (Eq. (8));
15     6. Lower threshold such that  $D_{\text{min}}$  holds;
16     7. Update  $f_i$  using this threshold;
17    $D_{i+1} = D_i \times D_{\text{min}}$ ;
18    $F_{i+1} = F_i \times f_i$ ; and remove those correctly classified
19   negative samples from the training set;
20   if  $F_{\text{target}} < F_i$  then
21     Evaluate the current cascaded classifier on the negative
22     images and add misclassified samples into the negative
23     training set;

```

---

with

$$Z^{(t+1)} = \sum_i u_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x}_i)).$$

Here  $\alpha_t = \log((1 - e_t)/(e_t))$  and  $e_t$  is the weighted error. We also use AsymBoost [5] re-weighting scheme (BGS LDA - scheme 2).

$$u_i^{(t+1)} = \frac{u_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \exp(y_i \log \sqrt{k})}{Z^{(t+1)}}, \quad (8)$$

with

$$Z^{(t+1)} = \sum_i u_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x}_i)) \exp(y_i \log \sqrt{k}).$$

Since BGS LDA based object detection framework has the same input/output as GSLDA based detection framework, we replace lines 2 – 10 in Algorithm 1 with Algorithm 2.

#### D. Training Time Complexity of BGS LDA

In order to analyze the complexity of the proposed system, we need to analyze the complexity of boosting and GSLDA training. Let the number of training samples in each cascade layer be  $N$ . For boosting, finding the optimal threshold of each feature needs  $O(N \log N)$ . Assume that the size of the feature set is  $M$  and the number of weak classifiers to be selected is  $T$ . The time complexity for training boosting classifier is  $O(MTN \log N)$ . The time complexity for GSLDA forward pass is  $O(NMT + MT^3)$ .  $O(N)$  is the time complexity for finding mean and variance of each features.  $O(T^2)$  is the time complexity for calculating correlation for each feature. Since, we have  $M$  features and the number of weak classifiers to be selected is  $T$ , the total time for complexity for GSLDA is  $O(NMT + MT^3)$ . Hence, the total time



Fig. 3. A random sample of face images for training.

TABLE I  
THE SIZE OF TRAINING AND TEST SETS USED ON THE SINGLE NODE CLASSIFIER.

#	data splits	faces/split	non-faces/split
Train	3	2000	2000
Test	2	2000	2000

complexity is  $O(\underbrace{MTN \log N}_{\text{weak classifier}} + \underbrace{NMT + MT^3}_{\text{GSLDA}})$ . Since,  $T$  is often small (less than 200) in cascaded structure, the term  $O(MTN \log N)$  often dominates. In other words, most of the computation time is spent on training weak classifiers.

### III. EXPERIMENTS

This section is organized as follows. The datasets used in this experiment, including how the performance is analyzed, are described. Experiments and the parameters used are then discussed. Finally, experimental results and analysis of different techniques are presented.

#### A. Face Detection with the GSLDA Classifier

Due to its efficiency, Haar-like rectangle features [1] have become a popular choice as image features in the context of face detection. Similar to the work in [1], the weak learning algorithm known as decision stump and Haar-like rectangle features are used here due to their simplicity and efficiency. The following experiments compare AdaBoost and GSLDA learning algorithms in their performances in the domain of face detection. For fast AdaBoost training of Haar-like rectangle features, we apply the pre-computing technique similar to [7].

1) *Performances on Single-node Classifiers*: This experiment compares single strong classifier learned using AdaBoost and GSLDA algorithms in their classification performance. The datasets consist of three training sets and two test sets. Each training set contains 2,000 face examples and 2,000 non-face examples (Table I). The dataset consists of 10,000 mirrored faces. The faces were cropped and rescaled to images of size  $24 \times 24$  pixels. For non-face examples, we randomly selected 10,000 random non-face patches from non-face images obtained from the internet. Fig. 3 shows a random sample of face training images.

For each experiment, three different classifiers are generated, each by selecting two out of the three training sets and the remaining training set for validation. The performance is measured by two different curves:- the test error rate and the classifier learning goal (the false alarm error rate on test set given that the detection rate on the validation set is fixed

at 99%). A 95% confidence interval of the true mean error rate is given by the t-distribution. In this experiment, we test two different approaches of GSLDA: forward-pass GSLDA and dual-pass (forward+backward) GSLDA. The results are shown in Fig. 4. The following observations can be made from these curves. Having the same number of learned Haar-like rectangle features, GSLDA achieves a comparable error rate to AdaBoost on test sets (Fig. 4(a)). GSLDA seems to perform slightly better with less number of Haar-like features ( $< 100$ ) while AdaBoost seems to perform slightly better with more Haar-like features ( $> 100$ ). However, both classifiers perform almost similarly within 95% confidence interval of the true error rate. This indicates that features selected using GSLDA classifier are as meaningful as features selected using AdaBoost classifier. From the curve, GSLDA with bi-directional search yields better results than GSLDA with forward search only. Fig. 4(b) shows the false positive error rate on test set. From the figure, both GSLDA and AdaBoost achieve a comparable false positive error rate on test set.

2) *Performances on Cascades of Strong Classifiers*: In this experiment, we used 5,000 mirrored faces from previous experiment. The non-face samples used in each cascade layer are collected from false positives of the previous stages of the cascade (bootstrapping). The cascade training algorithm terminates when there are not enough negative samples to bootstrap. For fair evaluation, we trained both techniques with the same number of weak classifiers in each cascade. Note that since dual pass GSLDA (forward+backward search) yields better solutions than the forward search in the previous experiment, we use dual pass GSLDA classifier to train a cascade of face detectors. We tested our face detectors on the low resolution faces dataset, MIT+CMU frontal face test set. The complete set contains 130 images with 507 frontal faces. In this experiment, we set the scaling factor to 1.2 and window shifting step to 1. The technique used for merging overlapping windows is similar to [1]. Detections are considered true or false positives based on the area of overlap with ground truth bounding boxes. To be considered a correct detection, the area of overlap between the predicted bounding box and ground truth bounding box must exceed 50%. Multiple detections of the same face in an image are considered false detections.

Figs. 5(a) and 5(b) show a comparison between the Receiver Operating Characteristic (ROC) curves produced by GSLDA classifier and AdaBoost classifier. In Fig. 5(a), the number of weak classifiers in each cascade stage is predetermined while in Fig. 5(b), weak classifiers are added to the cascade until the predefined objective is met. The ROC curves show that GSLDA classifier outperforms AdaBoost classifier at all false positive rates. We think that by adjusting the threshold to the AdaBoost classifier (in order to achieve high detection rates with moderate false positive rates), the performance of AdaBoost is no longer optimal. Our findings in this work are consistent with the experimental results reported in [5] and [7]. [7] used LDA weights instead of weak classifiers' weights provided by AdaBoost algorithm.

GSLDA not only performs better than AdaBoost but it is also much simpler. Weak classifiers learning (decision stumps) is performed only once for the given set of samples (unlike



AdaBoost where weak classifiers have to be re-trained in each boosting iteration). GSLDA algorithm sequentially selects decision stump whose output yields the maximum eigenvalue. The process continues until the stopping criteria are met. Note that given the decision stumps selected by GSLDA, any linear classifiers can be used to calculate the weight coefficients. Based on our experiments, using linear SVM (maximizing the minimum margin) instead of LDA also gives a very similar result to our GSLDA detector. We believe that using one objective criterion for feature selection and another criterion for classifier construction would provide a classifier with more flexibility than using the same criterion to select feature and train weight coefficients. These findings open up many more possibilities in combining various feature selection techniques with many existing classification techniques. We believe that a better and faster object detector can be built with careful design and experiment.

Haar-like rectangle features selected in the first cascade layer of both classifiers are shown in Fig. 7. Note that both classifiers select Haar-like features which cover the area around the eyes and forehead. Table II compares the two cascaded classifiers in terms of the number of weak classifiers and the average number of Haar-like rectangle features evaluated per detection window. Comparing GSLDA with AdaBoost, we found that GSLDA performance gain comes at the cost of a higher computation time. This is not surprising since the number of decision stumps available for training GSLDA classifier is much smaller than the number of decision stumps used in training AdaBoost classifier. Hence, AdaBoost classifier can choose a more powerful/meaningful decision stump. Nevertheless, GSLDA classifier outperforms AdaBoost classifier. This indicates that the classifier trained to maximize class separation might be more suitable in the domain where the distribution of positive and negative samples is highly skewed. In the next section, we conduct an experiment on BGS LDA.

### B. Face Detection with BGS LDA classifiers

The following experiments compare BGS LDA and different boosting learning algorithms in their performances for face detection. BGS LDA (weight scheme 1) corresponds to GSLDA classifier with decision stumps being re-weighted using AdaBoost scheme while BGS LDA (weight scheme 2) corresponds to GSLDA classifier with decision stumps being re-weighted using AsymBoost scheme (for highly skewed sample distributions). AsymBoost used in this experiment is from [5]. However, any asymmetric boosting approach can be applied here *e.g.* [23], [21].

1) *Performances on Single Node Classifiers:* The experimental setup is similar to the one described in previous section. The results are shown in figure 4. The following conclusions can be made from figure 4(c). Given the same number of weak classifiers, BGS LDA always achieves lower generalization error rate than AdaBoost. However, in terms of training error, AdaBoost achieves lower training error rate than BGS LDA. This is not surprising since AdaBoost has a faster convergence rate than BGS LDA. From the figure, AdaBoost

only achieves lower training error rate than BGS LDA when the number of Haar-like rectangle features  $> 50$ . Fig. 4(d) shows the false alarm error rate. The false positive error rate of both classifiers are quite similar.

2) *Performances on Cascades of Strong Classifiers:* The experimental setup and evaluation techniques used here are similar to the one described in Section III-A1. The results are shown in Fig. 5. Fig. 5(a) shows a comparison between the ROC curves produced by BGS LDA (scheme 1) classifier and AdaBoost classifier trained with the same number of weak classifiers in each cascade. Both ROC curves show that the BGS LDA classifier outperforms both AdaBoost and AdaBoost+LDA [7]. Fig. 5(b) shows a comparison between the ROC curves of different classifiers when the number of weak classifiers in each cascade stage is no longer predetermined. At each stage, weak classifiers are added until the predefined objective is met. Again, BGS LDA significantly outperforms other evaluated classifiers. Fig. 8 demonstrates some face detection results on our BGS LDA (scheme 1) detector.

In the next experiment, we compare the performance of BGS LDA (scheme 2) with other classifiers using asymmetric weight updating rule [5]. In other words, the asymmetric multiplier  $\exp(\frac{1}{N} y_i \log \sqrt{k})$  is applied to every sample before each round of weak classifier training. The results are shown in Fig. 6. Fig. 6(a) shows a comparison between the ROC curves trained with the same number of weak classifiers in each cascade stage. Fig. 6(b) shows the ROC curves trained with 99.5% detection rate and 50% false positive rate criteria. From both figures, BGS LDA (scheme 2) classifier outperforms other classifiers evaluated. BGS LDA (scheme 2) classifier also outperforms BGS LDA (scheme 1) classifier. This indicates that asymmetric loss might be more suitable in domains where the distribution of positive examples and negative examples is highly imbalanced. Note that the performance gain between BGS LDA (scheme 1) and BGS LDA (scheme 2) is quite small compared with the performance gain between AdaBoost and AsymBoost. Since, LDA takes the number of samples of each class into consideration when solving the optimization problem, we believe this reduces the performance gap between BGS LDA (scheme 1) and BGS LDA (scheme 2).

Table II indicates that our BGS LDA (scheme 1) classifier performs at a speed comparable to AdaBoost classifier. However, compared with AdaBoost+LDA, the performance gain of BGS LDA comes at the slightly higher cost in computation time. In terms of cascade training time, on a desktop with an Intel Core™ 2 Duo CPU T7300 with 4GB RAM, the total training time is less than one day.

As mentioned in [24], a more general technique for generating discriminating hyperplanes is to define the total within-class covariance matrix as

$$S_w = \sum_{x_i \in C_1} (x_i - \mu_1)(x_i - \mu_1)^T + \gamma \sum_{x_i \in C_2} (x_i - \mu_2)(x_i - \mu_2)^T, \quad (9)$$

where  $\mu_1$  is the mean of class 1 and  $\mu_2$  is the mean of class 2. The weighting parameter  $\gamma$  controls the weighted classification error. We have conducted an experiment on BGS LDA (scheme 1) with different value of  $\gamma$ , namely

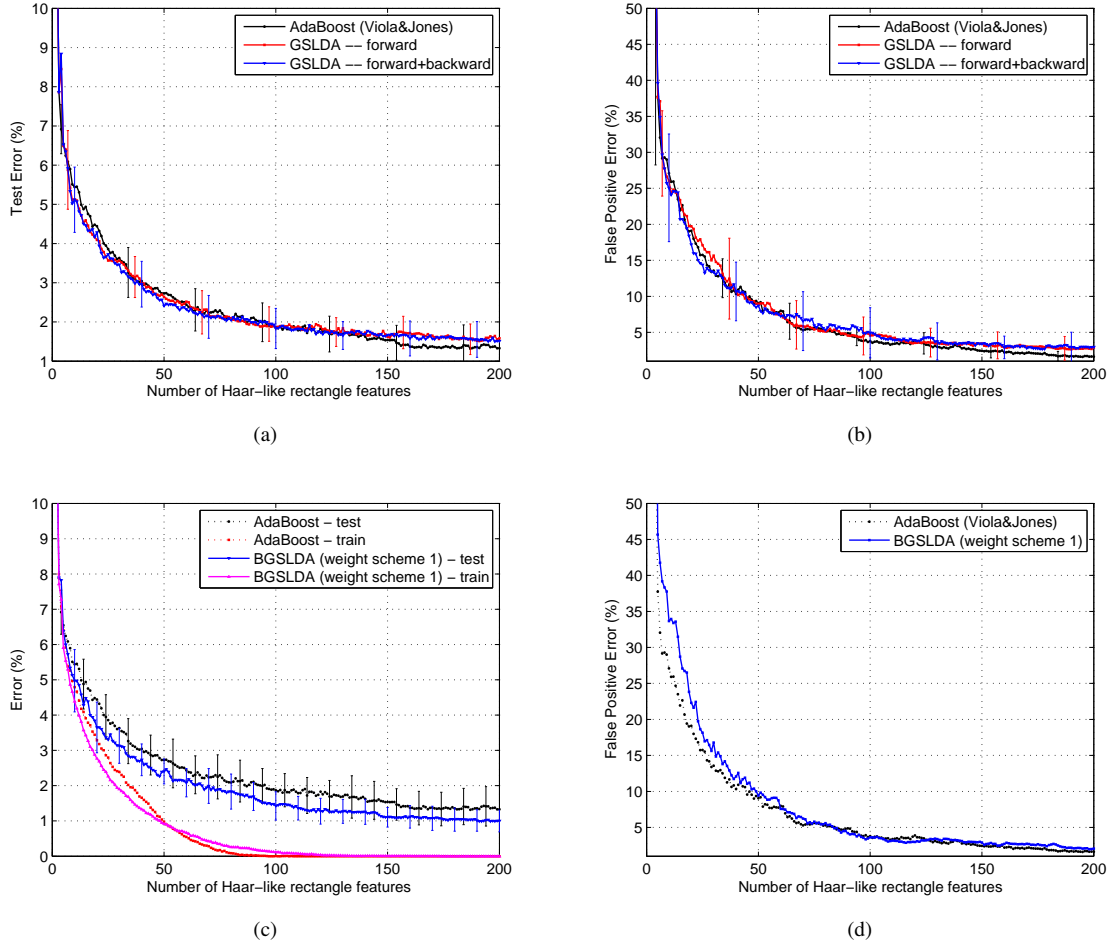


Fig. 4. See text for details (best viewed in color). (a) Comparison of test error rates between GSLDA and AdaBoost. (b) Comparison of false alarm rates on test set between GSLDA and AdaBoost. The detection rate on the validated face set is fixed at 99%. (c) Comparison of train and test error rates between BGSLDA (scheme 1) and AdaBoost. (d) Comparison of false alarm rates on test set between BGSLDA (scheme 1) and AdaBoost.

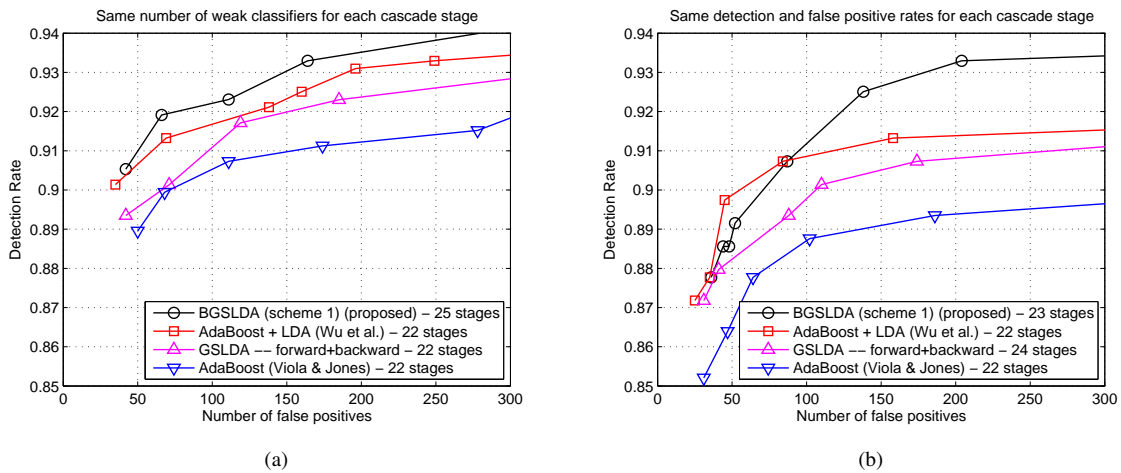


Fig. 5. Comparison of ROC curves on the MIT+CMU face test set (a) with the same number of weak classifiers in each cascade stage on AdaBoost and its variants. (b) with 99.5% detection rate and 50% false positive rate in each cascade stage on AdaBoost and its variants. BGSLDA (scheme 1) corresponds to GSLDA classifier with decision stumps being re-weighted using AdaBoost scheme.

$\gamma \in \{0.1, 0.5, 1.0, 2.0, 10.0\}$ . All the other experiment settings remain the same as described in the previous section. The

results are shown in Fig. 9. Based on ROC curves, it can be seen that *all configurations of BGSLDA classifiers outperform*



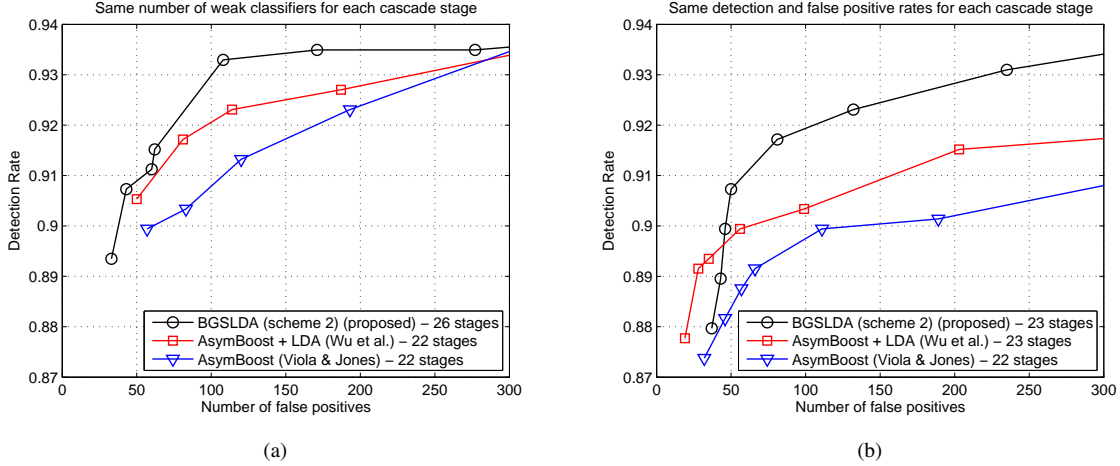


Fig. 6. Comparison of ROC curves on the MIT+CMU face test set (a) with the same number of weak classifiers in each cascade stage on AsymBoost and its variants. (b) with 99.5% detection rate and 50% false positive rate in each cascade stage on AsymBoost and its variants. BGSLDA (scheme 2) corresponds to GSLDA classifier with decision stumps being re-weighted using AsymBoost scheme.

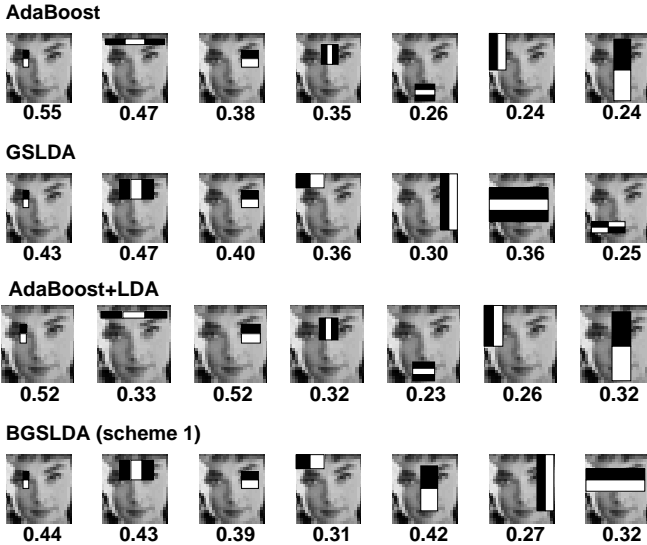


Fig. 7. The first seven Haar-like rectangle features selected from the first layer of the cascade. The value below each Haar-like rectangle features indicates the normalized feature weight. For AdaBoost, the value corresponds to the normalized  $\alpha$  where  $\alpha$  is computed from  $\log((1 - e_t)/e_t)$  and  $e_t$  is the weighted error. For LDA, the value corresponds to the normalized  $w$  such that for input vector  $x$  and a class label  $y$ ,  $w^T x$  leads to maximum separation between two classes.

*AdaBoost classifier at all false positive rates.* Setting  $\gamma = 1$  gives the highest detection rates when the number of false positives is larger than 200. Setting  $\gamma = 0.5$  performs best when the number of false positives is very small.

### C. Pedestrian Detection with GSLDA and BGSLDA Classifiers

In this section, we apply the proposed algorithm to pedestrian detection, which is considered a more difficult problem than face detection.

*1) Pedestrian Detection on the Daimler-Chrysler dataset with Haar-like Features:* In this experiment, we evaluate the

performance of our techniques on Daimler-Chrysler pedestrian dataset [25]. The dataset contains a set of extracted pedestrian and non-pedestrian samples which are scaled to size  $18 \times 36$  pixels. The dataset consists of three training sets and two test sets. Each training set contains 4,800 pedestrian examples and 5,000 non-pedestrian examples. Performance on the test sets is analyzed similarly to the techniques described in [25]. For each experiment, three different classifiers are generated. Testing all three classifiers on two test sets yields six different ROC curves. A 95% confidence interval of the true mean detection rate is given by the t-distribution. We conducted three experiments using Haar-like features trained with three different classifiers: AdaBoost, GSLDA and BGSLDA (scheme 1). The experimental setup is similar to the previous experiments.

Fig. 10 shows detection results of different classifiers. Again, the ROC curves show that LDA classifier outperforms AdaBoost classifier at all false positive rates. Clearly these curves are consistent with those on face datasets.

*2) Pedestrian Detection on the INRIA dataset with Covariance Features:* We also conduct experiments on INRIA pedestrian datasets. We compare the performance of our method with other state-of-the-art results. The INRIA dataset [26] consists of one training set and one test set. The training set contains 2,416 mirrored pedestrian examples and 1,200 non-pedestrian images. The pedestrian samples were obtained from manually labeling images taken at various time of the days and various locations. The pedestrian samples are mostly in standing position. A border of 8 pixels is added to the sample in order to preserve contour information. All samples are scaled to size  $64 \times 128$  pixels. The test set contains 1,176 mirrored pedestrian examples extracted from 288 images and 453 non-pedestrian test images.

Since, Haar-like features perform poorly on this dataset, we apply covariance features instead of Haar-like features [27], [18]. However, decision stump can not be directly applied since the algorithm is not applicable to multi-dimensional data. To overcome this problem, we apply LDA that projects a

TABLE II

COMPARISON OF NUMBER OF WEAK CLASSIFIERS. THE NUMBER OF CASCADE STAGES AND TOTAL WEAK CLASSIFIERS WERE OBTAINED FROM THE CLASSIFIERS TRAINED TO ACHIEVE A DETECTION RATE OF 99.5% AND THE MAXIMUM FALSE POSITIVE RATE OF 50% IN EACH CASCADE LAYER. THE AVERAGE NUMBER OF HAAR-LIKE RECTANGLES EVALUATED WAS OBTAINED FROM EVALUATING THE TRAINED CLASSIFIERS ON MIT+CMU FACE TEST SET.

method	number of stages	total number of weak classifiers	average number of Haar features evaluated
AdaBoost [1]	22	1771	23.9
AdaBoost+LDA [7]	22	1436	22.3
GSLDA	24	2985	36.0
BGSLDA (scheme 1)	23	1696	24.2
AsymBoost [5]	22	1650	22.6
AsymBoost+LDA [7]	22	1542	21.5
BGSLDA (scheme 2)	23	1621	24.9

**Algorithm 3** The algorithm for training multi-dimensional features.

```

1 foreach multi-dimensional feature do
2   1. Calculate the projection vector with LDA and project the
   multi-dimensional feature to 1D space;
3   2. Train decision stump classifiers to find an optimal
   threshold  $\theta$  using positive and negative training set;

```

multi-dimensional data onto a 1D space first. In brief, we stack covariance features and project them onto 1D space. Decision stumps are then applied as weak classifiers. Our training technique is different from [18]. [18] applied AdaBoost with weighted linear discriminant analysis (WLDA) as weak classifiers. The major drawback of [18] is a slow training time. Since, each training sample is assigned a weight, weak classifiers (WLDA) need to be trained  $T$  times, where  $T$  is the number of boosting iterations. In this experiment, we only train weak classifiers (LDA) once and store their projected result into a table. Because most of the training time in [18] is used to train WLDA, the new technique requires only  $\frac{1}{T}$  training time as that of [18]. After we project the multi-dimensional covariance features onto a 1D space using LDA, we train decision stumps on these 1D features. In other words, we replace line 4 and 5 in Algorithm 1 with Algorithm 3.

In this experiment, we generate a set of over-complete rectangular covariance filters and subsample the over-complete set in order to keep a manageable set for the training phase. The set contains approximately 45,675 covariance filters. In each stage, weak classifiers are added until the predefined objective is met. We set the minimum detection rate to be 99.5% and the maximum false positive rate to be 35% in each stage. The cascade threshold value is then adjusted such that the cascade rejects 50% negative samples on the training sets. Each stage is trained with 2,416 pedestrian samples and 2,500 non-pedestrian samples. The negative samples used in each stage of the cascades are collected from false positives of the previous stages of the cascades.

Fig. 11 shows a comparison of our experimental results on learning 1D covariance features using AdaBoost and GSLDA. The ROC curve is generated by adding one cascade level at a time. From the curve, GSLDA classifier outperforms AdaBoost classifiers at all false positive rates. The results seem to be consistent with our results reported earlier on face detection. On a closer observation, our simplified technique

performs very similar to existing covariance techniques [27], [18] at low false positive rates (lower than  $10^{-5}$ ). This method, however, seems to perform poorly at high false positive rates. Nonetheless, most real-world applications often focus on low false detections. Compared to boosted covariance features, the training time of cascade classifiers is reduced from *weeks* to *days* on a standard PC.

#### IV. CONCLUSION

In this work, we have proposed an alternative approach in the context of visual object detection. The core of the new framework is greedy sparse linear discriminant analysis (GSLDA) [2], which aims to maximize the class-separation criterion. On various datasets for face detection and pedestrian detection, we have shown that this technique outperforms AdaBoost when the distribution of positive and negative samples is highly skewed. To further improve the detection result, we have proposed a boosted version GSLDA, which combines boosting re-weighting scheme with decision stumps used for training the GSLDA algorithm. Our extensive experimental results show that the performance of BGSLDA is better than that of AdaBoost at a similar computation cost.

Future work will focus on the search for more efficient weak classifiers and on-line updating the learned model.

#### REFERENCES

- [1] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comp. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
- [2] B. Moghaddam, Y. Weiss, and S. Avidan, "Fast pixel/part selection with sparse eigenvectors," in *Proc. IEEE Int. Conf. Comp. Vis.*, Rio de Janeiro, Brazil, 2007.
- [3] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–28, 1998.
- [4] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake, "Computationally efficient face detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, Vancouver, 2001, vol. 2, pp. 695–700.
- [5] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Proc. Adv. Neural Inf. Process. Syst.* 2002, pp. 1311–1318, MIT Press.
- [6] S. Z. Li, K. L. Chan, and C. Wang, "Performance evaluation of the nearest feature line method in image classification and retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1335–1349, 2000.
- [7] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 369–382, 2008.
- [8] M.-T. Pham and T.-J. Cham, "Fast training and selection of Haar features using statistics in boosting-based face detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, Rio de Janeiro, Brazil, 2007.

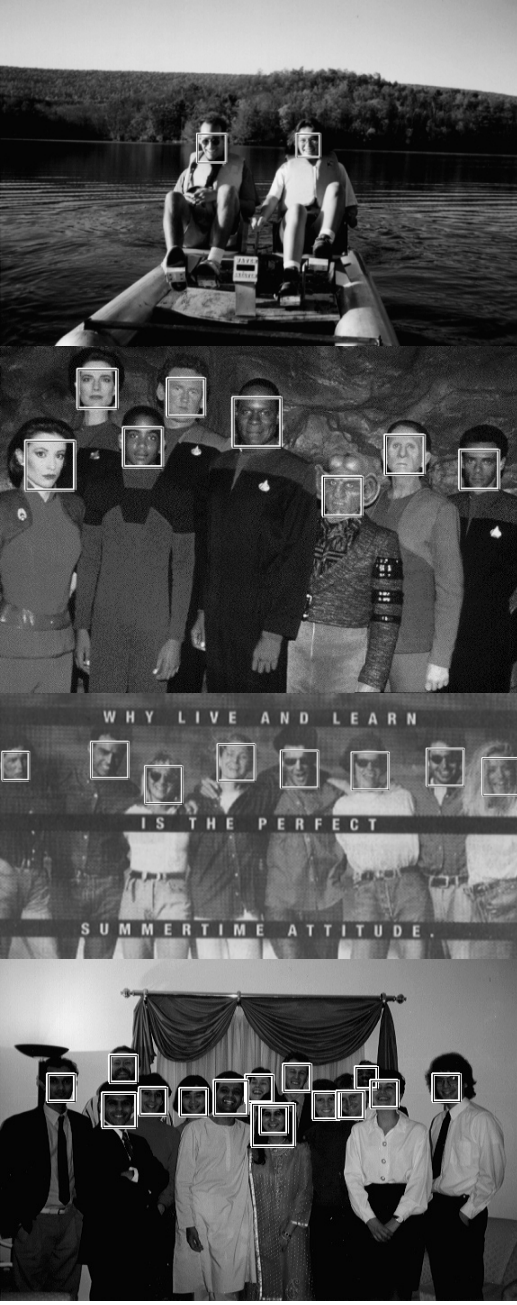


Fig. 8. Face detection examples using the BGSLDA (scheme 1) detector on the MIT+CMU test dataset. We set the scaling factor to 1.2 and window shifting step to 1 pixel. The technique used for merging overlapping windows is similar to [1].

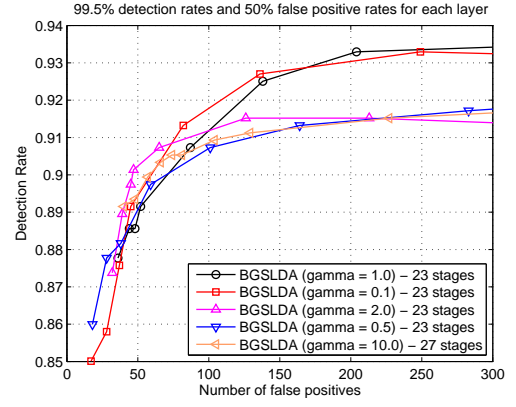


Fig. 9. Comparison of ROC curves with different values of  $\gamma$  in (9).

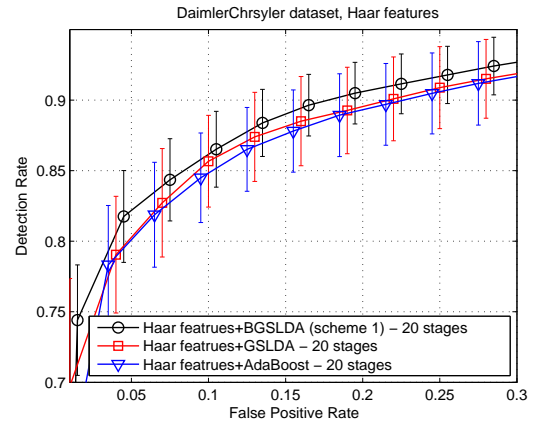


Fig. 10. Pedestrian detection performance comparison on the DaimlerChrysler pedestrian dataset [25].

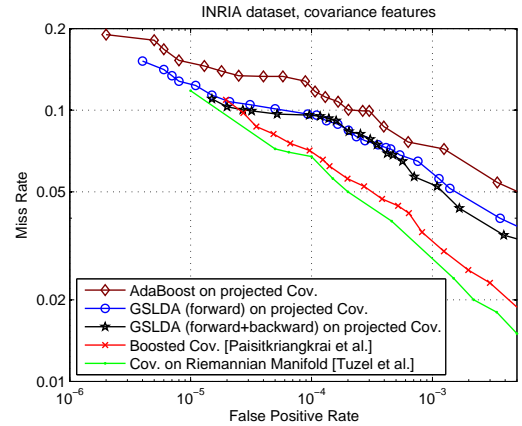


Fig. 11. Pedestrian detection performance comparison of 1D covariance features (projected covariance) trained using AdaBoost and GSLDA on the INRIA dataset [26].

- [9] C. Liu and H.-Y. Shum, "Kullback-Leibler boosting," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Madison, Wisconsin, June 2003, vol. 1, pp. 587–594.
- [10] S. Paisitkriangkrai, C. Shen, and J. Zhang, "Efficiently training a better visual detector with sparse Eigenvectors," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Miami, Florida, US, June 2009.
- [11] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, San Diego, CA, US, 2005, pp. 236–243.
- [12] M. T. Pham, V. D. D. Hoang, and T. J. Cham, "Detection with multi-exit asymmetric boosting," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Alaska, US, 2008, pp. 1–8.
- [13] A. R. Webb and D. Lowe, "The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis," *IEEE Trans. Neural Netw.*, vol. 3, no. 4, pp. 367–375, 1990.

- [14] B. Moghaddam, Y. Weiss, and S. Avidan, "Generalized spectral bounds for sparse l<sub>1</sub>da," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2006, pp. 641–648, ACM.
- [15] G. H. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 3rd edition, 1996.
- [16] T. Zhang, "Multi-stage convex relaxation for learning with sparse regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008, pp. 1929–1936.

- [17] A. Destrero, C. De Mol, F. Odone, and A. Verri, "A sparsity-enforcing method for learning face features," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 188–201, 2009.
- [18] S. Paisitkriangkrai, C. Shen, and J. Zhang, "Fast pedestrian detection using a cascade of boosted covariance features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1140–1151, 2008.
- [19] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
- [20] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1-3, pp. 225–254, 2002.
- [21] J. Leskovec, "Linear programming boosting for uneven datasets," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 456–463.
- [22] R. E. Schapire, "Theoretical views of boosting and applications," in *Proc. Int. Conf. Algorithmic Learn. Theory*, London, UK, 1999, pp. 13–25, Springer-Verlag.
- [23] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: Misclassification cost-sensitive boosting," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 1999, pp. 97–105.
- [24] T. Cooke and M. Peake, "The optimal classification using a linear discriminant for two point classes having known mean and covariance," *J. Multivariate Analysis*, vol. 82, pp. 379–394, 2002.
- [25] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1863–1868, 2006.
- [26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, San Diego, CA, 2005, vol. 1, pp. 886–893.
- [27] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on Riemannian manifolds," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, Minneapolis, MN, 2007.